Thesis for The Degree of Licentiate of Engineering

Software Concerns for Execution on Heterogeneous Platforms

Hugo Sica de Andrade

Division of Software Engineering
Department of Computer Science & Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden, 2018

Software Concerns for Execution on Heterogeneous Platforms

Hugo Sica de Andrade

Technical Report No 189L
ISSN 1652-876X
Department of Computer Science & Engineering
Division of Software Engineering
Chalmers University of Technology and University of Gothenburg
Gothenburg, Sweden

This thesis has been prepared using LaTeX.
Printed by Chalmers Reproservice,
Gothenburg, Sweden 2018.

To my father Hélio, my mother Gina, and my sister Débora.

Abstract

Context: Heterogeneous computing, i.e., computing performed on different types of execution units, such as CPUs, GPUs, FPGAs, has shown to be a feasible path towards higher performance and less energy consumption. Heterogeneous platforms are specialized on specific types of computation, e.g., parallel computing. However, this approach imposes a number of challenges on the software side. One of such challenges is related to software deployment, in which applications must be prepared to be executed in different target architectures. Further, the approach demands a robust inter-process communication solution, since these systems inherently distribute computation.
Objective: The objective of this thesis is twofold. First, to provide an overview of the state-of-the-art of software deployment on heterogeneous platforms, with emphasis to goals, concerns, challenges, and identification of topics of importance for further research. Second, to investigate the communication problem and propose a novel method that improves inter-process communication in distributed systems.
Method: Six papers were written as results of four studies: (i) a literature review in the form of a systematic mapping study on software deployment on heterogeneous platforms; (ii) a systematic evaluation of deployment methods in the context of a self-driving heavy vehicle; (iii) an investigation on data marshalling approaches and how they perform in the context of a cyber-physical system; and (iv) a novel message restructuring approach, also in the context of cyber-physical systems.
Results and Conclusions: The mapping study discussed the (i) concerns on the topic such as scheduling and software quality; the (ii) approaches available, such as frameworks; and the (iii) architecture solutions used, such as styles and principles. In the second study, we found that the performance decay is negligible when using sandboxed environments for deployment. In the third and fourth studies, we proposed and evaluated a data marshalling approach that decreases bandwidth consumption.
Future work: We intend to identify challenges that are currently faced in an industrial setting. In particular, a migration from a non-heterogenous platform to a heterogeneous platform can be studied in the context of a modern software development process, such as DevOps.

Keywords

Software Deployment, Software Architecture, Heterogeneous Platforms, Inter-process Communication

Acknowledgment

Big thanks to my supervisor, Ivica Crnkovic, for all his words of wisdom throughout the last years. He has been providing me not only with great guidance and opportunities on my professional path, but also with new and valuable perspectives on life. It is no wonder why everyone who has been in contact with Ivica gets amazed by his outstanding character.
Thanks to my co-supervisor, Christian Berger, with whom I have learned a lot about work processes, organizing and teaching. His very high dedication approach to being a researcher/engineer/teacher has shown me what hard work can get me in return.
Thanks to my fellow office sharing colleagues Federico Giaimo, Yue Kang and previously Hang Yin. My appreciation also goes out to all my Software Engineering division co-workers. I feel extremely privileged and honoured being able to work among such talented individuals.
I am very thankful to my parents and sister, who have managed to continue by my side even when I chose to be in different continents (multiple times).
Thanks to my north American host family, who have made me part of their family 14 years ago. This mental geographic triangulation has nothing against my feelings towards my family. I always have all them in my thoughts. Always.
Thanks to my girlfriend Anna, who has shown incredible support throughout my latest challenges. She has provided me with valuable encouragement and managed to find ways to kept me on track. Tack så mycket, gatinha!

List of Publications

Appended publications

This thesis is based on the following publications:

[A] H. Andrade, J. Schröder, I. Crnkovic "Software Deployment on Heterogeneous Platforms: A Systematic Mapping Study"
Submitted to the IEEE Transactions on Software Engineering journal (TSE).

[B] H. Andrade, I. Crnkovic "A Review on Software Architectures for Heterogeneous Platforms"
Proceedings of the 25th Asia-Pacific Software Engineering Conference (APSEC). Nara, Japan, 4-7 December, 2018.

[C] P. Masek, M. Thulin, H. Andrade, C. Berger, O. Benderius "Systematic Evaluation of Sandboxed Software Deployment for Real-time Software on the Example of a Self-Driving Heavy Vehicle"
Proceedings of the 19th IEEE International Conference on Intelligent Transportation Systems (ITSC). Rio de Janeiro, Brazil, 1-4 November, 2016.

[D] F. Giaimo, H. Andrade, C. Berger, I. Crnkovic "Improving Bandwidth Efficiency with Self-Adaptation for Data Marshalling on the Example of a Self-Driving Miniature Car"
Proceedings of the 9th European Conference on Software Architecture Workshops (ECSAW). Dubrovnik/Cavtat, Croatia, 7-11 September, 2015.

[E] H. Andrade, F. Giaimo, C. Berger, I. Crnkovic "Systematic Evaluation of Three Data Marshalling Approaches for Distributed Software Systems"
Proceedings of the Workshop on Domain-Specific Modeling (DSM @ SPLASH). Pittsburgh, USA, 27 October, 2015.

[F] H. Yin, F. Giaimo, H. Andrade, C. Berger, I. Crnkovic "Adaptive Message Restructuring using Model-Driven Engineering"
In: PLatifi S. (eds) Information Technology: New Generations. Advances in Intelligent Systems and Computing, vol 448. Springer, Cham, 2016.

Other publications

The following publications are not appended to this thesis, due to contents overlapping that of appended publications or contents not related to the thesis.

[a] H. Andrade "Investigating Software Deployment on Heterogeneous Platforms"
Proceedings of the 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). Venice, Italy, 5-8 April, 2016.

[b] H. Andrade, E. Almeida, I. Crnkovic "Architectural Bad Smells in Software Product Lines: An Exploratory Study"
Proceedings of the Working IEEE/IFIP Conference on Software Architecture Companion Volume (WICSA). Sydney, Australia, 7 April, 2014.

[c] D. Feitosa, A. Ampatzoglou, P. Avgeriou, F. J. Affonso, H. Andrade, K. R. Felizardo, E. Y. Nakagawa "Design Approaches for Critical Embedded Systems: A Systematic Mapping Study"
In: Damiani E., Spanoudakis G., Maciaszek L. (eds) Evaluation of Novel Approaches to Software Engineering. ENASE 2017. Communications in Computer and Information Science, vol 866. Springer, Cham, 2018.

[d] G. N. Rodrigues, A. Knauss, F. Guimaraes, G. Rodrigues, H. Andrade, J. Araujo, R. Ali "GoalD: A Goal-Driven Deployment Framework for Dynamic and Heterogeneous Computing Environments"
In submission to the Information and Software Technology journal (IST).

Personal Contribution

I was the main driver in Papers A and B, from the composition of the review protocol to writing the manuscripts. Throughout the study, I had multiple iterations with my supervisor, and part of the review process was aided by a colleague.
Paper C is derived from a master thesis that I closely co-supervised. I helped in the definition of goals, research design, literature review, and writing of the paper.
The studies leading to Papers D, E and F, were jointly conducted by the authors, who equally contributed to all phases of the research. From drawing sketches on the whiteboard to reviewing the text, we held several meetings and worked as a team.

Contents

Chapter 1

Introduction

The demands for computing performance continue to increase in science and engineering. A clear rise in the amount of data that is processed by computer systems is evident in multiple domains. The scenario is particularly challenging in the case of embedded systems, which are often limited in resources, as well as real-time and interfaces constraints [1].

In the past, the increasing requirements for hardware performance were fulfilled by (i) boosting the frequency of processing units (PUs) and/or by (ii) adding transistors onto processors. Since the frequency cannot be further increased with todays technology [2], performance is primarily boosted by an increased transistor count. However, as the number of transistors built on chips has reached several billions (c.f. Intel (Altera) Stratix 10 featuring over 30 billion transistors), it is increasingly difficult to make use of this many of them. On the software side, the added complexity in such platforms should be accounted for as soon as in the design phase, when different software deployment strategies may be modeled and analyzed.

One way to fulfill these high demands for performance is to consider a heterogeneous platform, i.e., a hardware platform consisting of different types of computational units and technologies. Heterogeneous platforms may contain, for instance, a combination of multi-core Central Processing Units (CPUs), Graphics Processing Units (GPUs) and Field-Programmable Gate Arrays (FPGAs), creating the impression of dedicated units that can be adapted to a wide range of application domains. These dedicated units can significantly increase the overall system's performance and energy management through, for instance, optimizing the workload distribution according to the types of data to be executed.

Accelerators such as GPUs and FPGAs are gaining popularity because they offer performance improvements in many applications. However, despite the fact that the difficulty in programming for such platforms is decreasing [3], there are multiple concerns that must be addressed in order to handle the inherent complexity of both hardware and software in such environments.

In this project, we focus on the software engineering side, based on the need to provide support for software development on heterogeneous platforms.

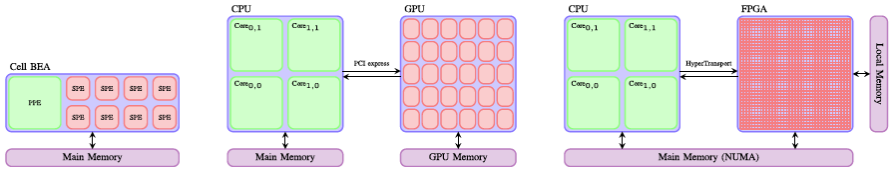
The remainder of this introduction section of thesis is organized as follows. In Section 1.1, we introduce the background. In Section 1.2, we describe the research goal and research questions of this thesis. In Section 1.3, we describe the research methodologies used in this research. In Section 1.4, we summarize the contributions of this thesis. In Section 1.5, we present the publications appended to this thesis. In Section 1.6, we describe the threats to validity of this work. Then, in Section 1.7, we discuss the conclusions. Finally, in Section 1.8, we present our intentions for future work.

1.1 Background

In this section, we provide the background for the main topics covered in this thesis: heterogeneous platforms (heterogeneous computing), software deployment, the automotive domain and inter-process communication.

1.1.1 Heterogeneous platforms

During our investigation, we discovered multiple studies that refer to the term "Heterogeneous platform" in different ways. Besides meaning different processors, we found that this term also refers to platforms containing processors of the same type, but with different capacities. For instance, a system that includes 2 CPUs with a different number of cores and/or clock frequencies is often called heterogeneous. Another situation in which the term is commonly found is when the types and further characteristics of the processors are omitted, being discussed only the difference in capacity of the PUs. For example, strictly combinatorial problems that consider a cost formula and a few performance attributes of the processors in order to determine the best deployment strategy. In this thesis, we adopted the definition used in [4]. The author defines heterogeneous computing as *complex systems composed of different kinds of processing units which use different processing paradigms and are designed for different types of tasks which work together in order to provide the best processing performance for diverse computing needs.* In this sense, we consider "heterogeneous platform" a hardware set consisting of at least two different types of processors that are specialized in different types of tasks.
An example of heterogeneous hardware architectures is shown in Figure 1.1 [2]. In Figure 1.1(a), the single-chip Cell Broadband Engine Architecture (CBEA) is depicted consisting of a traditional CPU core and eight single-instruction multiple data (SIMD) accelerator cores. Each core can run separate programs and communicate through a fast on-chip bus. Its main design criteria is to maximize performance while consuming minimum power. Figure 1.1(b) illustrates a GPU with 30 highly multi-threaded SIMD accelerator cores in combination with a standard multicore CPU. The GPU has superior bandwidth and computational performance. It is designed for high-performance graphics, where throughput of data is key. In Figure 1.1(c), a standard multi-core CPU is paired with an FPGA consisting of an array of logic blocks. FPGAs can also incorporate regular CPU cores on-chip, making it a heterogeneous chip by itself. FPGAs offer fully deterministic performance and are designed for high throughput, for example, in telecommunication applications.



(a) CBEA (b) CPU in combination with GPU (c) CPU in combination with FPGA

Figure 1.1: Example of heterogeneous hardware architectures as shown in [2]: (a) Cell Broadband Engine (heterogeneous chip), (b) a CPU in combination with a GPU, and (c) a CPU in combination with an FPGA.

1.1.2 Software deployment

The concept of deployment *also varies* according to the context in which the study is performed. For business research, it may refer to strategies for update releases of a mobile app. For technology research, deployment may refer to the tools that are used to facilitate and enable deployment, e.g., Docker [5]. For fundamental research, it may refer to the mathematical strategies to optimize load balance in a heterogeneous environment.

In the context of software engineering, *software deployment* comprises a set of activities resulting in a system that is available for use [6]. These activities can be very diverse and include a wide range of processes, such as users training, integration of new features into the existing system, the actual installation of software on the underlying hardware, etc. The scope of this project is limited to the activities involved in the process of installing software on hardware, including the decision about the units in which software components will be executed (component allocation). The activities of partitioning the software system into components and planning their execution on different processing units are considered in this work. Further, we focus on deployment from the software perspective rather than from the hardware perspective. We do not discuss organizational/business issues around deployment, the stakeholders involved in the process, or the training of prospective users.

As we conducted this work, we realized that the activities performed in the typical deployment stage are heavily influenced by activities in previous stages in the software process. For instance, we learned that one common way to realize deployment onto heterogeneous platforms is by using a development framework, which needs to be applied as soon as in the architecture phase. For this reason, we extend the concept of deployment to include all activities that are relevant throughout the software engineering process to successfully execute software onto a heterogeneous platform.

One generic representation is shown in Figure 1.2, where a deployment scenario is depicted. On the hardware side, there is a heterogeneous platform consisting of an FPGA, N CPUs and M GPUs that are available for processing data, and these units have interfaces with different types of sensors and actuators. The software is decomposed into components that can be deployed according to different configurations, while the following assumptions might be relevant: (i) vehicle data instructions might execute in a shorter time on the FPGA when compared to CPUs or GPUs, however programming for FPGAs might

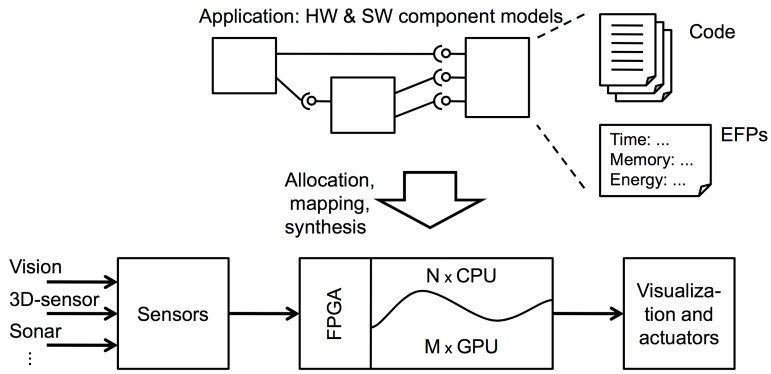


Figure 1.2: Generic modelling scenario of cyber-physical systems, depicting software deployment on a heterogeneous platform [7].

be complex and more time consuming; (ii) two dependent applications might be executed faster in different executing units, however the communication between them might be compromised by the available bandwidth; (iii) allocating components and running image processing applications in parallel on a single executing unit (e.g., GPU) might be less complex, but also compromise energy efficiency. Other aspects may also be considered, such as the impact of the technology used to encapsulate processes for software deployment, or the underlying environment in which the application is executed.

1.1.3 Automotive domain & Inter-process communication

In the automotive domain, a modern vehicle comprises of a set of Electronic Control Units (ECUs) interconnected via a network to provide functionalities such as stability control, measurement of essential fluids and infotainment systems. In the context of a self-driving vehicles, these ECUs are also responsible for handling data obtained by sensing the surrounding environment. These types of systems are also called *cyber-physical systems*, given such interaction between the software/electronic capabilities and the physical world, these types of systems.
From the software perspective, self-driving vehicles require a middleware that bridges high-level and low-level data and ideally enables seamless interaction between the different hardware components that are part of the system. In the context of this research, we use the OpenDaVINCI software environment [8], which provides middleware functionalities to enable communication between all the distributed software modules in the context of a self-driving vehicle. The software comprises of interfaces to the supported hardware components and devices, like cameras, sensors and laser scanners.
OpenDaVINCI is currently used in three different settings, as follows.

ReVeRe. In the facilities provided by Chalmers University of Technologys vehicle laboratory "ReVeRe" (Resource for Vehicle Research), multiple projects are conducted in collaborations between academia and industrial partners.

The main goal of this research setting is to provide the infrastructure that allows researchers to create breakthrough technology in the area of self-driving vehicles and collision avoidance. The lab equipment comprises, among others, an SUV (Volvo XC90), and a truck tractor (Volvo FH16).

Miniature vehicles. Within the academic context, miniature vehicles on a 1:10 scale are used as (i) experimental platforms for our research, and (ii) educational platforms in university courses. These vehicles contain hardware components including ESC, sensors, and camera that are used to perform a number of activities without human intervention. For instance, in the context of the B.Sc. course, the students work on a software implementation that enables the miniature vehicle to perform lane-following, parking, and overtaking.

Simulation environment. OpenDaVINCI includes a simulation environment in which it is possible to experiment self-driving capability algorithms with the help of generated sensor data. It is possible to model virtual tracks and add obstacles that emulate real-life scenarios in a driving context.

In addition to the hardware capabilities on handling data, we have observed that one influencing aspect regarding the overall performance of embedded systems is related to the communication between different components (i.e. data marshaling). Because software deployed on heterogeneous platforms is distributed in essence, the exchange of messages between different components/processes play a key role in determining whether or not satisfactory results in terms of performance will be achieved.
Especially in the context of embedded systems (or cyber-physical systems), it is important to define an effective communication strategy because resources are typically limited and several applications are safety-critical (e.g., self-driving vehicles).

1.2 Research Goal

The main goal of this thesis is *to investigate software concerns for execution on heterogeneous platforms*. By concerns, we mean topics that practitioners must be aware of and ultimately address when deploying software on hardware containing more than one type of processor. Throughout the conduction of the research, we discovered that the concept of deployment varies according to the context in which the study is performed. In this work, we consider the software engineering perspective, which covers methods, processes, and techniques that enable and lead to the execution of software on heterogeneous platforms.
Based on the research goal, we formulated three research questions, as follows.

RQ1: What are the main concerns and approaches in software deployment on heterogeneous platforms?

As the first step towards understanding the area of research, we aimed to investigate the state-of-the-art of software deployment on heterogeneous platforms, focusing on the main concerns and approaches that can be found in the literature. The purpose of RQ1 is to obtain an overview of the body of

knowledge in the area by revealing concerns and approaches that are relevant when deploying software on heterogeneous platforms.

RQ2: What is the impact of different deployment strategies on non-functional properties when designing embedded systems?

With RQ2, we aimed to study the impact of different deployment strategies on the systems performance. The idea is to measure the possible trade-offs in utilizing common deployment strategies, such as sandboxed environments. In some critical domains (e.g., self-driving vehicles), performance is crucial and must be guaranteed throughout the execution processes, despite the limited resources available on such embedded systems.

RQ3: How can the communication between different computational resources be improved?

Further in the context of self-driving vehicles, we have observed that one of the main issues that may hinder performance is the inter-process communication between computational resources. With RQ3, we aimed to investigate the problem and propose solutions to this issue, which commonly represents a bottleneck to such systems due to the increased amounts of data that are transmitted.
In summary, RQ1 and RQ2 focus on *software deployment*, while RQ3 focuses on *inter-process communication*.

1.3 Research Methodology

We used three different research methodologies to address the formulated research questions, as follows.

1.3.1 Systematic literature reviews

To address RQ1, we conducted a systematic literature review in the form of a mapping study. Mapping Studies differ from classic Systematic Literature Reviews in their broadness and depth [9,10]. Instead of rigorously searching, analyzing and assessing studies, selected information is extracted from the primary studies in order to obtain an overview of the current state-of-the-art of research in a particular field.
We aimed at performing a systematic approach to increase reliability of the study and enable reproducibility in the future. The search included popular academic databases and followed a set of predefined inclusion and exclusion criteria. After the selection of studies from the libraries, we performed the snowballing procedure [11] to also cover related papers. The review selected 146 primary studies, and therefore collected a large amount of data to be analyzed and discussed. We followed the standard rigorous procedure for mapping studies, and complemented with a bottom-up approach to find similar common characteristics of the studies. Finally, we provided different classifications in order to achieve a better understanding of the area.
To address RQ2, we conducted a smaller literature review followed by two controlled experiments. The review followed the same principles as in the

guidelines for performing systematic mapping studies, although it only contained on research question and searched in only one (major) digital library. We also performed the snowballing procedure to cover related papers that were possibly neglected when selecting studies from the digital library.

1.3.2 Controlled experiments

Controlled experiments help to investigate a testable hypothesis where one or more independent variables are manipulated to measure their effect on one or more dependent variables [12].

The first controlled experiment *on desk* was conducted in the context of RQ2, in which we measured the scheduling precision and I/O performance of sample applications when deployed on different environments. Through a sequence of controlled steps, the sample applications were executed in four different execution environments, consisting of an alternation of (i) executing the sample applications natively or sandboxed within a Docker container and (ii) executing the sample applications on a target system with a vanilla or a real-time enabled kernel.

Finally, the second experiment used a real-world application that was at the time deployed on a self-driving truck. It was designed in a way that the findings from the first experiment could be validated.

For RQ3, we conducted controlled experiments to assess our approach compared with our existing solutions using evaluation scenarios. We studied the existing approaches and assessed their performances under various, application-independent conditions. We ran an experiment that collected data in a systematic way, using fixed parameters and different message attribute combinations.

1.3.3 Design science

The design science methodology focuses on the design and investigation of artifacts in a given context [13]. In concrete terms, the methodology comprises of an iterative process in which researchers engage in three main activities: (i) problem identification and opportunities representation; (ii) development of solutions; and (iii) evaluations of the proposed solutions in a given context in order to determine whether the solutions effectively address the problem.

In the context of RQ3, we followed the design science approach [14] to identify the problem of bandwidth limitation in cyber-physical systems. From our experience, this aspect represented a bottleneck in our systems, thus representing opportunities for improvement. Then, we proposed a solution referring to data marshalling approaches, in which the bandwidth consumption is decreased by optimizing the composition of messages to be transmitted. The approach was implemented in the context of self-driving cars, including both simulation environments and real-world scenarios. Finally, we evaluated our proposed approach by conducting experiments to compare the performance of existing approaches against ours. In the case of our message restructuring proposal, we formally validated the model's correctness using a widely known model-checking technology.

1.4 Contributions

In this section, we present a summary of the four research contributions, followed by their relation to the papers included in this thesis, and my personal contribution to them. These contributions are the outcome of our investigations on the topics that were previously described, aiming to answer the research questions. We started by conducting a systematic literature review in the form of a mapping study to obtain an overview of the area, and leading to Contribution 1. This study revealed several approaches that can be used for deploying software on heterogeneous platforms. Then, from observing our current projects in the lab, we came across the possibility to adopt a widely known deployment tool, which led to Contribution 2. Also from observations on our cyber-physical systems domain, we explored the area of data marshalling approaches, leading to Contributions 3 and 4.
The relation between research questions, contributions, papers, and their main topics is shown in Table 1.1.

1. An overview of the main concerns and approaches of software deployment on heterogeneous platforms;

2. A systematic evaluation of sandboxed software deployment strategies;

3. A data marshalling approach for reducing bandwidth consumption;

4. A message restructuring approach for improving resource usage.

Table 1.1: Relation between contributions, research questions, papers, and main topics of the papers.

Contributions	Questions	Papers	Main topic
Contribution 1	RQ1	Paper A Paper B	Deployment
Contribution 2	RQ2	Paper C	
Contribution 3 Contribution 4	RQ3	Paper D Paper E Paper F	Communication

1.4.1 Contribution 1: An overview of the main concerns and approaches of software deployment on heterogeneous platforms

We systematically searched and analyzed the literature in order to obtain an overview of the research area, discovering gaps and trends. We considered papers indexed by trusted libraries in computer science and followed a predefined process to formulate the research questions, conduct the research, screen the papers, and extract data from them.

This study led to two papers, one focusing on the main concerns and approaches (Paper A [15]), and another focusing on architectural aspects (Paper B [16]) of software deployment on heterogeneous platforms.

Personal contribution: I was the main driver of all steps in this study. The main idea was formulated by Ivica Crnkovic, and the study was conducted mainly by me, with the help of Jan Schröder in the paper screening process. Weekly meetings were held between Ivica and me to align the focus of the review, resolve disagreements, and plan future steps.

1.4.2 Contribution 2: A systematic evaluation of sandboxed software deployment strategies

In this study, we aimed at systematically evaluating the influence of sandboxed execution environments for applications in the automotive domain. We were particularly interested in studying the impact on two quality attributes of the system: scheduling precision and input/output performance. We elected Docker as the deployment tool to be evaluated, due to: (i) its growing popularity, and (ii) our interests in adopting Docker for our ReVeRe project for self-driving vehicles.

This study led to Paper C [17].

Personal contribution: This study was based on the master thesis by Philip Masek and Magnus Thulin at the Chalmers | University of Gothenburg [18], whom I co-supervised with Christian Berger. The concept, scope, and setting of the project were designed by Christian and I. The experiments were conducted by Philip and Magnus under my close supervision, and facilitated in the lab by Ola Benderius. The paper was jointly written by all co-authors.

1.4.3 Contribution 3: A data marshalling approach for reducing bandwidth consumption

We proposed and evaluated our concept of self-adaptive data marshalling, that aimed at reducing bandwidth usage. The main idea was to improve inter-process communication by only sending the difference (i.e., the *delta*) between the current message and the previous. The communication protocol is transparent and was designed in the publish/subscribe architectural pattern. Our approach was then evaluated against well-established data marshalling approaches: LCM [19] and Google Protobuf [20].

This study led to two papers: one focusing on the proposal of our approach (Paper D [21], and another focusing on its evaluation against existing approaches (Paper E [22]).

Personal contribution: This study was conducted jointly by Federico Giaimo, Christian Berger and I. From the conceptual discussions to writing the papers, we equally shared the workload and held frequent discussions to reach agreements and define the next steps.

1.4.4 Contribution 4: A message restructuring approach for improving resource usage

We continued to explore inter-process communication issues by proposing a model-based approach for adaptive message restructuring. The approach aims at reducing message latency by dynamically restructuring messages according to both design- and runtime properties. Design-time parameters included, e.g., the composition of fields in a message; while runtime parameters included, e.g., message transmission latency, timeout and message arrival/drop rate.

This study led to Paper F [23].

Personal contribution: This study was conducted jointly by Hang Yin, Federico Giaimo, Christian Berger and I. We equally shared all the workload related to designing the concept, conducting the validation and writing the paper.

1.5 Publications

In this section, we list the main publications related to this thesis. The complete version of the papers can be found in Chapters 2 to 7. They have been reformatted in order to comply with the layout of this thesis.

Paper A

Software Deployment on Heterogeneous Platforms: A Systematic Mapping Study
H. Andrade, J. Schröder, I. Crnkovic
Submitted to the IEEE Transactions on Software Engineering journal (TSE).

The main goal in Paper A was to investigate the state-of-the-art of software deployment on heterogeneous platforms. We systematically searched and analyzed the literature in order to obtain an overview of the research area, discovering gaps and trends. We considered papers indexed by trusted libraries in computer science and followed a pre-defined process to formulate the research questions, conduct the research, screen the papers, and extract data from them.

Abstract: *Context:* Multiple types of processing units (e.g., CPUs, GPUs and FPGAs) can be used jointly to achieve better performance in computational systems. However, these units are built with fundamentally different characteristics and demand attention especially towards software deployment. *Objective:* The goal of this work is to summarize the state-of-the art of software deployment on heterogeneous platforms. We provide an overview of the research area by searching for and categorizing relevant studies, as well as discussing gaps and trends of the field. We are interested in the main concerns (RQ1) and the approaches used (RQ2) when deploying software on heterogeneous platforms. *Method:* In order to achieve our goal, we performed a systematic mapping study, which refers to a method for reviewing literature with basis on predefined search strategies and a multi-step selection process. *Results:* We

selected and analyzed 146 primary studies from multiple sources, and found that the area of research is dominated by solution proposals. The majority of the studies discussed concerns about scheduling, the quality of the software, and its architecture. A large number of studies focused on the problem of scheduling tasks and processes. We found approaches that are applied at different binding times (i.e., design time, runtime, orthogonal). *Conclusion:* The evaluation of the proposed solutions in an industrial context are missing. Also, the proposed methods have not been evaluated in development processes. Most of the methods address a particular concern, or a few concerns, while there is a lack of a holistic approach.

Paper B

A Review on Software Architectures for Heterogeneous Platforms
H. Andrade, I. Crnkovic
Proceedings of the 25th Asia-Pacific Software Engineering Conference (APSEC). Nara, Japan, 4-7 December, 2018.

The systematic mapping study covered a large area of research, so we decided to split the results into Papers A and Paper B. While Paper A focused on *concerns* and *approaches*, Paper B focused on *software architecture*.

Abstract: The increasing demands for computing performance have been a reality regardless of the requirements for smaller and more energy efficient devices. Throughout the years, the strategy adopted by industry was to increase the robustness of a single processor by increasing its clock frequency and mounting more transistors so more calculations could be executed. However, it is known that the physical limits of such processors are being reached, and one way to fulfill such increasing computing demands has been to adopt a strategy based on heterogeneous computing, i.e., using a heterogeneous platform containing more than one type of processor. This way, different types of tasks can be executed by processors that are specialized in them. Heterogeneous computing, however, poses a number of challenges to software engineering, especially in the architecture and deployment phases. In this paper, we conduct an empirical study that aims at discovering the state-of-the-art in software architecture for heterogeneous computing, with focus on deployment. We conduct a systematic mapping study that retrieved 28 studies, which were critically assessed to obtain an overview of the research field. We identified gaps and trends that can be used by both researchers and practitioners as guides to further investigate the topic.

Paper C

Systematic Evaluation of Sandboxed Software Deployment for Real-time Software on the Example of a Self-Driving Heavy Vehicle
P. Masek, M. Thulin, H. Andrade, C. Berger, O. Benderius

Proceedings of the 19th IEEE International Conference on Intelligent Transportation Systems (ITSC). Rio de Janeiro, Brazil, 1-4 November, 2016.

In Paper C, we aimed at systematically evaluating the influence of sandboxed execution environments for applications from the automotive domain. We were particularly interested in studying the impact on two quality attributes of the system: Scheduling precision and input/output performance.

Abstract: Companies developing and maintaining *software-only* products like web shops aim for establishing persistent links to their software running in the field. Monitoring data from real usage scenarios allows for a number of improvements in the software life-cycle, such as quick identification and solution of issues, and elicitation of requirements from previously unexpected usage. While the processes of continuous integration, continuous deployment, and continuous experimentation using sandboxing technologies are becoming well established in said software-only products, adopting similar practices for the automotive domain is more complex mainly due to real-time and safety constraints. In this paper, we systematically evaluate sandboxed software deployment in the context of a self-driving heavy vehicle that participated in the 2016 Grand Cooperative Driving Challenge (GCDC) in The Netherlands. We measured the system's scheduling precision after deploying applications in four different execution environments. Our results indicate that there is no significant difference in performance and overhead when sandboxed environments are used compared to natively deployed software. Thus, recent trends in software architecting, packaging, and maintenance using *microservices* encapsulated in sandboxes will help to realize similar software and system engineering for cyber-physical systems.

Paper D

Improving Bandwidth Efficiency with Self-Adaptation for Data Marshalling on the Example of a Self-Driving Miniature Car

F. Giaimo, H. Andrade, C. Berger, I. Crnkovic

Proceedings of the 9th European Conference on Software Architecture Workshops (ECSAW). Dubrovnik/Cavtat, Croatia, 7-11 September, 2015.

In Paper D, we proposed and evaluated our concept of self-adaptive data marshalling, that aimed at reducing bandwidth usage. The main idea was to improve inter-process communication by only sending the difference (i.e., the *delta*) between the current message and the previous. The communication protocol is transparent and designed in the publish/subscribe architectural pattern.

Abstract: Publish/subscribe communication is a common architectural design pattern in component-based software systems used in many of today's cyber-physical systems to exchange information between distributed software components. These systems typically deal with an increased number of data transfers, with a risk of lacking resources. Our recent domain analysis for a lane-following algorithm of a self-driving miniature car unveiled that the

actual "information increment" between two subsequently sent packets is often small. Such scenario enables possibilities for a more efficient data exchange by avoiding redundant and/or unnecessary information transfer. In this paper, we propose and evaluate our concept for "self-adaptive data marshalling" that transparently adapts data types in messages to be exchanged by analyzing the actual information increment. The approach could reduce the bandwidth usage by more than 50% in comparison to the current approach, and by approximately 33% compared to the use of the general-purpose compression library zlib.

Paper E

Systematic Evaluation of Three Data Marshalling Approaches for Distributed Software Systems

H. Andrade, F. Giaimo, C. Berger, I. Crnkovic

Proceedings of the Workshop on Domain-Specific Modeling (DSM @ SPLASH). Pittsburgh, USA, 27 October, 2015.

As a follow-up from Paper D, we studied data marshalling approaches further by systematically evaluating three approaches: Google Protobuf, LCM, and our self-adaptive *delta* approach. We selected these two popular approaches as a way to further assess the efficiency of our proposed approach.

Abstract: Cyber-physical systems like robots and self-driving vehicles comprise complex software systems. Their software is typically realized as distributed agents that are responsible for dedicated tasks like sensor data handling, sensor data fusion, or action planning. The modular design allows a flexible deployment as well as algorithm encapsulation to exchange software modules where needed. The distributed software exchanges data using a data marshalling layer to serialize and deserialize data structures between a sending and receiving entity. In this article, we are systematically evaluating Google Protobuf, LCM, and our self-adaptive delta marshalling approach by using a generic description language, of which instances are composed at runtime. Our results show that Google Protobuf performs well for small messages composed mainly by integral field types; the self-adaptive data marshalling approach is efficient if four or more fields of type double are present, and LCM outperforms both when a mix of many integral and double fields is used.

Paper F

Adaptive Message Restructuring using Model-Driven Engineering

H. Yin, F. Giaimo, H. Andrade, C. Berger, I. Crnkovic

In: PLatifi S. (eds) Information Technology: New Generations. Advances in Intelligent Systems and Computing, vol 448. Springer, Cham, 2016.

In Paper F, we continued to explore inter-process communication issues by proposing a model-based approach for adaptive message restructuring. The approach aims at reducing message latency by dynamically restructuring messages

according to both design- and runtime properties. Design-time parameters included, e.g., the composition of fields in a message; while runtime parameters included, e.g., message transmission latency, timeout and message arrival/drop rate.

Abstract: Message exchange between distributed software components in cyber-physical systems is a frequent and resource-demanding activity. Existing data description languages simply map user-specified messages literally to the system implementation creating the data stream that is exchanged between the software components; however, our research shows that the exchanged information is often redundant and would allow for runtime optimization. In this paper, we propose a model-based approach for adaptive message restructuring. Taking both design-time properties and runtime properties into account, we propose to dynamically restructure user-specified messages to achieve better resource usage (e.g., reduced latency). Our model-based workflow also includes formal verification of adaptive message restructuring in the presence of complex data flow. This is demonstrated by an automotive example.

1.6 Threats to Validity

In this section, we provide an overview of the threats to validity of this work. They are structured according to the approach shown in [24], which classifies the validity into Construct, Internal, External, and Reliability. Detailed threats to validity to each included paper are discussed in their respective chapters.

1.6.1 Construct validity

The literature study leading to RQ1 covered a broad research area. The data extraction process was extensive and manually conducted, thus subject to interpretation. Further, the granularity of the selected categories, and consequently answers to questions and conclusions drawing were based on our judgment. As an attempt to reduce bias, two researchers performed the inclusion/exclusion process individually and then discussed the results, and in several iterations refined the categories. Disagreements were solved by a third researcher. We used common principles to guide our research, such as keeping the same or similar abstraction level of the categories, the orthogonality of the categories, and the completeness of the categories.

For RQ2, the study was realized within an established middleware to ensure a high degree of software correctness and completeness, meeting the design requirements for real-time systems. The experiments were conducted in close supervision of expert practitioners that validated both the research questions and the research procedures.

For RQ3, the design of the evaluation followed the regulations of the actual self-driving vehicles competition. The design of the studies used a generic message representation that can be systematically defined, allowing for a scenario-independent analysis of the performance of the evaluated approaches. In Paper F, we designed the properties of our validation model based on our experience from previous automotive projects.

1.6.2 Internal validity

In order to minimize the internal validity threats in addressing RQ1, we conducted standard systematic mapping study methods that included a predefined set of inclusion/exclusion criteria, pilot searches, and the calibration of the search string. Standard methods were also followed in RQ2, in which we systematically compared our current implementation and our proposed method with existing methods that could be used alternatively.

For RQ2, the execution of the sample applications was carried out by a script to ensure precise reproducibility of the experiments; all peripherals such as networking are detached; and data was collected via serial communication to limit additional load to the system.

For RQ3, we based our work on the guidelines for *design science* [25] in the development of the self-driving vehicle platform.

1.6.3 External validity

The results of the study leading to RQ2 can be applied to time-sensitive applications in respect to the hardware and software used. The hardware we used is industrial grade, making the experiment reproducible and the results relatable to similar contexts. The evaluation of our self-adaptive data marshalling approach against Google Protobuf and LCM can be considered as relevant as both other approaches are widely used and have shown the applicability in the domain of cyber-physical systems. Thus, the findings presented in this study have an impact on the design of such systems.

Regarding RQ3, our results were obtained by using the simulation. While this allows to conduct our study in a repeatable way, the expected savings might differ in reality as the manufacturing quality of the different sensors as well as environmental factors can influence the volatility of the data resulting in significant differences between consecutive packets.

In paper F, we considered the automotive domain as evaluation context, and only runtime properties need to be further evaluated using case studies from further domains.

1.6.4 Reliability

We rigorously followed well-established guidelines for conducting systematic literature reviews in order to minimize the reliability threat in RQ1. The concepts, goals, and directions were specified in a review protocol, and complied by all researchers involved. Frequent meetings were held to solve disagreements and answer questions that rose individually. Extensive information on the process and all data collected are available online to enable verification and possible reproducibility.

For RQ3, the range for the floats used in the evaluation was inspired by applications in the self-driving vehicles domain. As the goal for the delta approach was to address high frequency data exchanges, the motivation for the increment values was due to the small numeric difference between values in consecutive packets.

1.7 Conclusion

This thesis presents the motivation, procedures, and findings of our research conducted in the area of software deployment on heterogeneous platforms. The overall goal of the Ph.D. project is to understand the role of software deployment in achieving certain non-functional properties when a heterogeneous platform is available. Particularly on this licentiate thesis, we focused on two main topics: *deployment* and *communication*.
Under deployment, we conducted an extensive systematic literature study to obtain an overview of the area by "mapping out" the field of research, obtaining domain knowledge, and identifying gaps that indicated possibilities for future investigation. We identified several concerns and approaches for software deployment on heterogeneous platforms. Further on the deployment topic, we systematically evaluated a widely known technology for deploying sandboxed software in the context of cyber-physical systems.
Regarding the communication topic, we performed studies to understand how communication is typically done between software components and proposed a communication protocol that enables reduced bandwidth consumption. We compared the results using our approach with well-known data marshalling approaches, and pointed the cases in which each approach offers better performance. Finally, we proposed a novel model-based approach based on the restructuring of the messages, enabling the content of the communication packets to be reorganized. The approach saves bandwidth consumption by optimizing the use of each fixed-sized packet, thus reducing the number of packets that are transmitted.

1.8 Future Work

As future work, we intend to continue investigating software deployment on heterogeneous platforms by considering the following possibilities:

Exploring industrial settings. Especially with the popularization of artificial intelligence techniques, heterogeneous platforms can help in improving the performance of software systems in industry. It would be relevant to investigate how heterogeneous platforms can be useful in the context of modern, typically industrial software development processes, such as DevOps.

Studying the migration to heterogeneous platforms. In the context of cyber-physical systems, it would be possible to study the feasibility of adopting a heterogeneous computing strategy on our self-driving vehicles' project. The study would be able to identify the advantages and drawbacks of deploying software onto heterogeneous platforms in the context of a cyber-physical system.

Performing evaluation studies. As discovered in the mapping study, there are very few studies evaluating existing techniques, while the vast majority of them proposes solutions to a given problem. One could, for instance, setup an experiment to observe a variety of quality attributes when using different frameworks for heterogeneous computing (e.g., OpenCL, CUDA, OpenMP).